CHAPITRE 6

LES TABLEAUX

1.Problématique :

Jusque là, et avec les structures de données simple qu'on a déjà définies (entier, réel, caractère,…), on n'est capable de définir que des variables de types simples qui, à un instant donné, ne peut contenir qu'une seule valeur, et l'affectation d'une nouvelle valeur détruit l'ancienne.

Si par exemple, on voulait représenter les moyennes annuelle de 50 étudiants, la déclaration de 50 variables de type Réel est alors absurde, il sera plus commode si on avait une structure qui puisse englober toutes ces données. On parlera alors du ***type Tableau.***

2. Définition :

Un tableau est une structure qui regroupe plusieurs éléments de même type, chaque élément est repéré par un ou plusieurs indices avec lesquels on pourra accéder à sa valeur ou lui en affecter une nouvelle. Chaque indice est compris entre une borne inférieure et une borne supérieure définie lors de la déclaration du tableau.

3. Les tableaux unidimensionnels :

Un tableau unidimensionnel est aussi appelé vecteur est constitué d'un nombre finie de cases contigus et situé en mémoire centrale.

Un tableau est caractérisé par :

- son nom
- sa taille (borne inférieure et borne supérieure connues à l'avance)
- ses éléments : chaque élément est défini par son type et son contenu ; on accède à un élément du tableau à l'aide de son indice.

3.1 Représentation Algorithmique:

```
Variables
Nom_tab : tableau[borne_inf…bone_sup] de type d'élément
```

On peut aussi définir un type tableau comme suit :

```
Type
Tab =tableau [borne_inf…bone_sup] de type d'élément
Variables
T : Tab
```

Exemple :

Type

Moyenne : tableau [1..5] de réel

Variable

T : Moyenne

3.2 Accès à un élément d'un tableau:

L'accès à un élément d'un tableau se fait à l'aide de l'indice de cet élément qui est compris entre borne_inf et borne_sup de la manière suivante :

Nom_tab[indice]

Exemple :

Variables

T : Moyenne

Par exemple

T=

10.5	5.5	18	9.5	15.5

Pour accéder à la 3ème moyenne qui est 18 on fait T[3], pou accéder à la 5ème moyenne du tableau on fait T[5] ainsi de suite.

3.3 Opérations de base sur un tableau:

Remplissage d'un tableau :

Le remplissage d'un tableau se fait de deux manière différente soit en le remplissant élément par élément comme suit T[1] ←val1, T[2] ←val2, etc… ; soit à l'aide d'une boucle et on utilise généralement la boucle « Pour » parce que la taille du tableau est connue à l'avance c'est la façon la plus appropriée pour remplir un tableau.

Exercice d'application 1 :

Ecrire une procédure qui permet de saisir un tableau de 10 entiers au clavier.

Correction :

```
Type
Tab :tableau [1..10]d'entier
    Procédure Remplir(var t :Tab)
       Variables
         i : entier
      Début
        Pour i de 1 à 10 Faire
          Ecrire("donner un entier ")
          Lire(t[i])
        Fin Pour
      Fin
```

Remarque :

Comme le contenu du tableau va être modifié on doit mette le mot var devant le nom du tableau.

Affichage d'un tableau :

Comme la saisie du tableau, l'affichage d'un tableau se fait élément par élément et se fait généralement à l'aide de la boucle « Pour ».

Exercice d'application 2 :

Ecrire une procédure qui permet d'afficher le tableau déjà rempli dans l'exercice 1.

Correction :

```
Procédure Afficher(t : Tab)
   Variables
   i : entier
   Début
        Pour i de 1 à 10 Faire
          Ecrire(t[i])
        Fin Pour
   Fin
```

Exercice d'application 3 :

Ecrire une fonction qui retourne la somme des éléments d'un tableau de n entiers avec n<=10.

Correction :

```
Fonction Somme(t : tab, n : entier) : entier
    Variables
     i, s: entier
    Début
      s←0
        Pour i de 1 à n Faire
          s←s+t[i]
        Fin Pour
       Somme←s
     Fin
```

Exercice d'application 4 :

Ecrire une procédure qui permet d'afficher seulement les éléments strictement positifs d'un tableau de n entier.

Correction :

```
Procédure Afficher(t : Tab, n : entier)
    Variables
     i : entier
     Début
        Pour i de 1 à n Faire
             Si (t[i] >0) alors
                Ecrire(t[i])
             Fin Si
         Fin Pour
      Fin
```

3.4 Recherche dans un tableau:

Il s'agit de rechercher un élément saisi à partir du clavier, dans un tableau. Dès qu'on le trouve ce n'est plus la peine de continuer le parcours du tableau ; on doit sortir et afficher un message d'existence.

Deux façons de faire :

- Soit faire une recherche séquentielle,
- Soit faire une recherche plus optimisée dite recherche dichotomique et dans ce cas, il y a des précautions et des actions préalables à faire.

Recherche Séquentielle :

Là il s'agit de parcourir le tableau élément par élément et le comparer à l'élément rechercher s'il est différent on le compare à l'élément suivant du tableau et ainsi de suite jusqu'à ce qu'on trouve l'élément recherché ou que l'on atteigne la fin tu tableau.

Exercice d'application 5

Ecrire une procédure qui permet de rechercher un élément x dans un tableau de n entier si l'élément est trouver la procédure affiche le message « élément trouver à la position y » ou y et la position de la première occurrence de x dans le tableau.

Correction

```
Procédure Recherche(t : Tab, n : entier, x :entier)
    Variables
     i : entier
    Début
       i←1
        Tant que ((i <= n) et (t[i] # x) )Faire
          i←i+1
        Fin Tant que
        Si (i > n) alors
          Ecrire("l'élément n'existe pas")
       Sinon
          Ecrire("l'élément existe à la position", i)
       Fin Si
     Fin
```

Recherche par Dichotomie

Pour pouvoir appliquer la recherche dichotomique, il faut que le tableau soit **Trié.**

Il s'agit d'une recherche qui consiste à réduire le temps de recherche d'un élément X dans un tableau T qui doit être obligatoirement *trié*.

On commence par comparer X avec le contenu de l'élément du milieu : si X est inférieur à cette valeur alors on va continuer la recherche dans la partie gauche du tableau T avec modification de la borne supérieure, sinon on continuera la recherche dans partie droite du tableau avec modification de la valeur de la borne inférieure. On s'arrêtera quand X serait égale à la valeur du milieu ou quand on dépasse les bornes du tableau.

Exercice d'application 6

Ecrire une procédure qui recherche un élément x dans un tableau de n entiers en utilisant la recherche par dichotomie cette procédure va afficher la position de la première occurrence de

x dans le tableau si l'élément ne pas trouver la procédure affiche le message « élément introuvable ».

Correction

```
Procédure Recherchedichotomie(t : Tab, n : entier, x : entier)
   Variable
    debut, fin, milieu :entier

    Début
      debut←1
      fin←n
      trouve←faux
        Répeter
         milieu←(debut+fin) div 2
          Si (t[milieu]=x) alors
             trouve← vrai
          Sinon
               Si (t[milieu] < x) alors
                  debut← milieu+1
              Sinon
                  fin ←milieu-1
             Fin Si
         Fin Si
       Jusqu'à ((trouve=vrai )ou (debut > fin))
      Si ( trouve=vrai) alors
         Ecrire("l'élément existe à la position", milieu)
      Sinon
         Ecrire("l'élément n'existe pas")
      Fin Si
     Fin
```

3.5 Les Algorithmes de Tri:

L'opération de trier consiste à ordonner un ensemble d'éléments selon un **critère** bien déterminé (appelé aussi **clé**) et suivant une **relation d'ordre** (croissant ou décroissant).

Dans ce qui suit on va s'intéresser aux 3 techniques de tri qui sont : tri par Sélection, tri par Insertion et tri à Bulles.

Tri par Sélection

- ***Principe***

Le principe consiste à sélectionner l'élément dont la valeur est la plus basse (minimum), puis échanger la valeur de cet élément avec le premier élément du tableau. Le traitement doit

se poursuivre en cherchant le minimum parmi ceux qui restent : à partir du deuxième élément du tableau, puis faire l'échange avec le deuxième élément jusqu'à atteindre T [n-1] et T[n].

- ***Algorithme***

```
Algorithme Tri_Sélection
   Constante
     n : entier
   Type
     Tab_Entier = Tableau[1..n] de Entier
    Variable
      min, j, temp, i : entier
      T : Tab_Entier
    Début
      Pour i de 1 à (n-1) Faire
        min ← i
          Pour j de (i+1) à n Faire
             Si (T[j] < T[min]) Alors
               min ← j
             FinSi
          FinPour
        Temp ← T[min]
        T[min] ← T[i]
        T[i]  ← temp
      FinPour
    Fin
```

Exercice d'application 7 :

Réalisez la trace du tri par sélection sur le tableau suivant :

6	8	2	1	5

1ère itération

1	8	2	6	5

2ème itération

1	2	8	6	5

3ème itération

1	2	5	6	8

4ème itération

1	2	5	6	8

Tri par Insertion :

- ***Principe :***

L'idée du tri par Insertion est la suivante : on suppose que l'on ait un tableau trié jusqu'à l'élément i-1; pour créer un tableau trié jusqu'à i il suffit d'insérer l'élément i à la bonne position dans la partie déjà trié ; après insertion le tableau est bien trié jusqu'à l'élément i.

- ***Algorithme :***

```
Algorithme Tri_Insertion
  Constante
     n : entier
   Type
     Tab_Entier = Tableau[1..n] de     Entier
   Variable
      i, j, aux : entier
     T : Tab_Entier
    Début
      Pour i de 2 à n Faire
        aux ← T[i]
        j ← (i-1)
         Tant que ( j > 0 ) et ( T[j] > aux ) Faire
            T[j+1] ←T[j]
            j← j-1
         Fin Tant que
       T[j+1] ←aux
      Finpour
    Fin
```

Exercice d'application 8 :

Réalisez la trace du tri par Insertion sur le tableau suivant :

9	4	7	6	1

1ère itération

4	9	7	6	1

2ème itération

4	7	9	6	1

3ème itération

4	6	7	9	1

4ème itération

1	4	6	7	9

Tri à Bulles :

- ***Principe :***

Cette technique de tri consiste à balayer tout le tableau en comparant les éléments adjacents et en les échangeant s'ils ne sont pas dans le bon ordre. Un seul passage ne déplacera un élément donné que d'une position, mais en répétant le processus jusqu'à ce qu'aucun échange ne soit nécessaire.

- ***Algorithme :***

```
Algorithme Tri_Bulles
    Constante
      n : entier
    Type
     Tab_Entier = Tableau[1..n] de Entier
    Variable
      Permute : Booléen
      i, temp : entier
     T : Tab_Entier
     Début
        Répéter
          Permute← faux
          Pour i de 2 à n Faire
             Si (T[i-1] > T[i]) Alors
                 Permute←vrai
                 temp ←T[i-1]
                 T[i-1]←T[i]
                 T[i] ←temp
                 FinSi
            FinPour
         Jusqu'à (Non(Permute))
```

```
Fin
```

Exercice d'application 8 :

Réalisez la trace du tri par Insertion sur le tableau suivant :

6	4	2	1	4

1ère itération

4	2	1	4	6

2ème itération

2	1	4	4	6

3ème itération

1	2	4	4	6

4ème itération

1	2	4	4	6

4. Les tableaux à deux dimensions : Les Matrices

4.1 Définition:

Un tableau à deux dimensions appelé également matrice est une structure de données permettant d'organiser des informations de même type en lignes et en colonnes. Il est caractérisé donc par son nombre de lignes et son nombre de colonnes.

4.2 Représentation Algorithmique:

```
Type

Matrice=tableau[1..Nbligne, 1..Nbcolone] de type_élément

Variables

Mat : Matrice
```

Remarques :

- L'accès à un élément de la matrice ne peut se faire qu'avec deux indices : un élément est identifié par son numéro de ligne et son numéro de colonne.
- Si M est la matrice, **M[I,J]** désigne l'élément de M situé à la $I^{ème}$ Ligne et à la $J^{ème}$ Colonne.

4.3 Opérations de base sur les Matrices:

Remplissage d'une Matrice :

Le remplissage d'une matrice est assez proche du remplissage d'un tableau unidirectionnel sauf que dans ce cas on doit utiliser 2 boucles imbriquées une pour les lignes et l'autre pour les colonnes.

Exercice d'application 9 :

Ecrire une procédure qui permet de remplir une matrice de n lignes et m colonnes d'entiers avec n<=100 etm<=100.

Correction :

```
Type
Matrice=tableau [1..100,1..100]
Procédure  Remplissage(var mat :Matrice, n, m : entier)
     Variables
       i, j : entier
     Début
        Pour i de 1 à n Faire
            Pour j de 1 à m Faire
              Ecrire("donner un entier")
              Lire(mat[i,j])
           Fin Pour
        Fin Pour
    Fin
```

Affichage d'une Matrice :

Exercice d'application 10 :

Ecrire une procédure qui permet d'afficher une matrice de n lignes et m colonnes d'entiers.

Correction :

```
Procédure  Affichage(var mat :Matrice, n, m : entier)
    Variables
     i, j : entier
      Début
         Pour i de 1 à n Faire
               Pour j de 1 à m Faire
                  Ecrire(mat[i,j])
              Fin Pour
          Fin Pour
        Fin
```

Exercice d'application 11 :

Ecrire une procédure qui permet d'afficher tout les éléments d'une matrice qui sont supérieur à 10.

Correction :

```
Procédure  Affichage1(var mat :Matrice, n, m : entier)
        Variables
         i, j : entier
        Début
           Pour i de 1 à n Faire
                Pour j de 1 à m Faire
                   Si (mat[i,j] >10) Faire
                       Ecrire(mat[i,j])
                   Fin Si
                Fin Pour
            Fin Pour
        Fin
```

Exercice d'application 12 :

Ecrire une procédure qui permet de faire la somme de deux matrices A et B de taille nXm.

```
Procédure Somme(A,B : mat, var C : mat)
      Variables
         i, j : entier
      Début
         Pour i de 1 à n Faire
            Pour j de 1 à m Faire
                C[i, j]=A[i, j]+B[i, j]
           Fin Pour
         Fin Pour
      Fin
```